

# 1 Static Semantics

$\boxed{\Theta; \Delta; \Gamma \vdash e : t}$    Typing rules for expressions

$$\frac{}{\Theta; \Delta; \cdot, x : t \vdash x : t} \text{TY\_VAR\_LIN}$$

$$\frac{x : t \in \Delta}{\Theta; \Delta; \cdot \vdash x : t} \text{TY\_VAR}$$

$$\frac{\begin{array}{l} \Theta; \Delta; \Gamma \vdash e : t \\ \Theta; \Delta; \Gamma', x : t \vdash e' : t' \end{array}}{\Theta; \Delta; \Gamma, \Gamma' \vdash \mathbf{let } x = e \mathbf{ in } e' : t'} \text{TY\_LET}$$

$$\frac{}{\Theta; \Delta; \cdot \vdash () : \mathbf{unit}} \text{TY\_UNIT\_INTRO}$$

$$\frac{\begin{array}{l} \Theta; \Delta; \Gamma \vdash e : \mathbf{unit} \\ \Theta; \Delta; \Gamma' \vdash e' : t \end{array}}{\Theta; \Delta; \Gamma, \Gamma' \vdash \mathbf{let } () = e \mathbf{ in } e' : t} \text{TY\_UNIT\_ELIM}$$

$$\frac{}{\Theta; \Delta; \cdot \vdash \mathbf{true} : \mathbf{bool}} \text{TY\_BOOL\_TRUE}$$

$$\frac{}{\Theta; \Delta; \cdot \vdash \mathbf{false} : \mathbf{bool}} \text{TY\_BOOL\_FALSE}$$

$$\frac{\begin{array}{l} \Theta; \Delta; \Gamma \vdash e : \mathbf{!bool} \\ \Theta; \Delta; \Gamma' \vdash e_1 : t' \\ \Theta; \Delta; \Gamma' \vdash e_2 : t' \end{array}}{\Theta; \Delta; \Gamma, \Gamma' \vdash \mathbf{if } e \mathbf{ then } e_1 \mathbf{ else } e_2 : t} \text{TY\_BOOL\_ELIM}$$

$$\frac{}{\Theta; \Delta; \cdot \vdash k : \mathbf{int}} \text{TY\_INT\_INTRO}$$

$$\frac{}{\Theta; \Delta; \cdot \vdash el : \mathbf{elt}} \text{TY\_ELT\_INTRO}$$

$$\frac{\begin{array}{l} \Theta; \Delta; \cdot \vdash v : t \\ v \neq l \cdot f \end{array}}{\Theta; \Delta; \cdot \vdash \mathbf{Many } v : \mathbf{!}t} \text{TY\_BANG\_INTRO}$$

$$\frac{\begin{array}{l} \Theta; \Delta; \Gamma \vdash e : \mathbf{!}t \\ \Theta; \Delta, x : t; \Gamma' \vdash e' : t' \end{array}}{\Theta; \Delta; \Gamma, \Gamma' \vdash \mathbf{let Many } x = e \mathbf{ in } e' : t'} \text{TY\_BANG\_ELIM}$$

$$\frac{\begin{array}{l} \Theta; \Delta; \Gamma \vdash e : t \\ \Theta; \Delta; \Gamma' \vdash e' : t' \end{array}}{\Theta; \Delta; \Gamma, \Gamma' \vdash (e, e') : t \otimes t'} \text{TY\_PAIR\_INTRO}$$

$$\frac{\begin{array}{l} \Theta; \Delta; \Gamma \vdash e_{12} : t_1 \otimes t_2 \\ \Theta; \Delta; \Gamma', a : t_1, b : t_2 \vdash e : t \end{array}}{\Theta; \Delta; \Gamma, \Gamma' \vdash \mathbf{let } (a, b) = e_{12} \mathbf{ in } e : t} \text{TY\_PAIR\_ELIM}$$

$$\begin{array}{c}
\frac{\Theta \vdash t' \text{ Type} \quad \Theta; \Delta; \Gamma, x : t' \vdash e : t}{\Theta; \Delta; \Gamma \vdash \mathbf{fun} \, x : t' \rightarrow e : t' \multimap t} \quad \text{TY\_LAMBDA} \\
\\
\frac{\Theta; \Delta; \Gamma \vdash e : t' \multimap t \quad \Theta; \Delta; \Gamma' \vdash e' : t'}{\Theta; \Delta; \Gamma, \Gamma' \vdash e \, e' : t} \quad \text{TY\_APP} \\
\\
\frac{\Theta, fc; \Delta; \Gamma \vdash e : t}{\Theta; \Delta; \Gamma \vdash \mathbf{fun} \, fc \rightarrow e : \forall fc. t} \quad \text{TY\_GEN} \\
\\
\frac{\Theta \vdash f \text{ Cap} \quad \Theta; \Delta; \Gamma \vdash e : \forall fc. t}{\Theta; \Delta; \Gamma \vdash e[f] : t[f/fc]} \quad \text{TY\_SPC} \\
\\
\frac{\Theta; \Delta, g : t \multimap t'; \cdot, x : t \vdash e : t'}{\Theta; \Delta; \cdot \vdash \mathbf{fix} \, (g, x : t, e : t') : t \multimap t'} \quad \text{TY\_FIX}
\end{array}$$

## 2 Dynamic Semantics

$$\boxed{\langle \sigma, e \rangle \rightarrow \text{StepsTo}} \quad \text{operational semantics}$$

$$\frac{}{\langle \sigma, \mathbf{let} \, () = () \mathbf{in} \, e \rangle \rightarrow \langle \sigma, e \rangle} \quad \text{OP\_LET\_UNIT}$$

$$\frac{}{\langle \sigma, \mathbf{let} \, x = v \mathbf{in} \, e \rangle \rightarrow \langle \sigma, e[x/v] \rangle} \quad \text{OP\_LET\_VAR}$$

$$\frac{}{\langle \sigma, \mathbf{if} \, (\mathbf{Many} \, \mathbf{true}) \mathbf{then} \, e_1 \mathbf{else} \, e_2 \rangle \rightarrow \langle \sigma, e_1 \rangle} \quad \text{OP\_IF\_TRUE}$$

$$\frac{}{\langle \sigma, \mathbf{if} \, (\mathbf{Many} \, \mathbf{false}) \mathbf{then} \, e_1 \mathbf{else} \, e_2 \rangle \rightarrow \langle \sigma, e_2 \rangle} \quad \text{OP\_IF\_FALSE}$$

$$\frac{}{\langle \sigma, \mathbf{let} \, \mathbf{Many} \, x = \mathbf{Many} \, v \mathbf{in} \, e \rangle \rightarrow \langle \sigma, e[x/v] \rangle} \quad \text{OP\_LET\_MANY}$$

$$\frac{}{\langle \sigma, \mathbf{let} \, (a, b) = (v_1, v_2) \mathbf{in} \, e \rangle \rightarrow \langle \sigma, e[a/v_1][b/v_2] \rangle} \quad \text{OP\_LET\_PAIR}$$

$$\frac{}{\langle \sigma, (\mathbf{fun} \, fc \rightarrow v)[f] \rangle \rightarrow \langle \sigma, v[fc/f] \rangle} \quad \text{OP\_FRAC\_CAP}$$

$$\frac{}{\langle \sigma, \mathbf{fix} \, (g, x : t, e : t') \, v \rangle \rightarrow \langle \sigma, e[x/v][g/\mathbf{fix} \, (g, x : t, e : t')] \rangle} \quad \text{OP\_APP\_FIX}$$

$$\frac{}{\langle \sigma, (\mathbf{fun} \, x : t \rightarrow e) \, v \rangle \rightarrow \langle \sigma, e[x/v] \rangle} \quad \text{OP\_APP\_LAMBDA}$$

$$\frac{\langle \sigma, e \rangle \rightarrow \langle \sigma', e' \rangle}{\langle \sigma, C[e] \rangle \rightarrow \langle \sigma, C[e'] \rangle} \quad \text{OP\_CONTEXT}$$

$$\frac{\langle \sigma, e \rangle \rightarrow \mathbf{err}}{\langle \sigma, C[e] \rangle \rightarrow \mathbf{err}} \quad \text{OP\_CONTEXT\_ERR}$$

$$\frac{0 \leq k_1, k_2 \quad l \text{ fresh}}{\langle \sigma, \mathbf{matrix} \ k_1 \ k_2 \rangle \rightarrow \langle \sigma + \{l \mapsto_1 M_{k_1, k_2}\}, l \cdot 1 \rangle} \quad \text{OP\_MATRIX}$$

$$\frac{}{\langle \sigma + \{l \mapsto_1 m_{k_1, k_2}\}, \mathbf{free} \ l \cdot 1 \rangle \rightarrow \langle \sigma, () \rangle} \quad \text{OP\_FREE}$$

$$\frac{}{\langle \sigma + \{l \mapsto_f m_{k_1, k_2}\}, \mathbf{share} \ l \cdot f \rangle \rightarrow \langle \sigma + \{l \mapsto_{\frac{1}{2}f} m_{k_1, k_2}\} + \{l \mapsto_{\frac{1}{2}f} m_{k_1, k_2}\}, (l \cdot \frac{1}{2}f, l \cdot \frac{1}{2}f) \rangle} \quad \text{OP\_SHARE}$$

$$\frac{f \leq 1 \quad v \equiv l \cdot \frac{1}{2}f}{\langle \sigma + \{l \mapsto_{\frac{1}{2}f} m_{k_1, k_2}\} + \{l \mapsto_{\frac{1}{2}f} m_{k_1, k_2}\}, \mathbf{unshare} \ v \ v \rangle \rightarrow \langle \sigma + \{l \mapsto_f m_{k_1, k_2}\}, l \cdot f \rangle} \quad \text{OP\_UNSHARE\_EQ}$$

$$\frac{l \neq l'}{\langle \sigma + \{l \mapsto_{\frac{1}{2}f} m_{k_1, k_2}\} + \{l' \mapsto_{\frac{1}{2}f} m'_{k_1, k_2}\}, \mathbf{unshare} \ (l \cdot \frac{1}{2}f) \ (l' \cdot \frac{1}{2}f') \rangle \rightarrow \mathbf{err}} \quad \text{OP\_UNSHARE\_NEQ}$$

$$\frac{\begin{array}{l} \sigma' \equiv \sigma + \{l_1 \mapsto_{fc_1} m_{1k_1, k_2}\} + \{l_2 \mapsto_{fc_2} m_{2k_2, k_3}\} \quad v_1 \equiv l_1 \cdot f_1 \quad v_2 \equiv l_2 \cdot f_2 \\ v_3 \equiv l_3 \cdot 1 \end{array}}{\langle \sigma' + \{l_3 \mapsto_1 m_{3k_1, k_3}\}, \mathbf{gemm} \ v_1 \ v_2 \ v_3 \rangle \rightarrow \langle \sigma' + \{l_3 \mapsto_1 (m_1 \ m_2 + m_3)_{k_1, k_3}\}, ((v_1, v_2), v_3) \rangle} \quad \text{OP\_GEMM\_MATCH}$$

$$\frac{\begin{array}{l} k_2 \neq k'_2 \\ \sigma' \equiv \sigma + \{l_1 \mapsto_{fc_1} m_{1k_1, k_2}\} + \{l_2 \mapsto_{fc_2} m_{2k'_2, k_3}\} \\ v_1 \equiv l_1 \cdot f_1 \quad v_2 \equiv l_2 \cdot f_2 \quad v_3 \equiv l_3 \cdot 1 \end{array}}{\langle \sigma' + \{l_3 \mapsto_1 m_{1k_1, k_3}\}, \mathbf{gemm} \ v_1 \ v_2 \ v_3 \rangle \rightarrow \mathbf{err}} \quad \text{OP\_GEMM\_MISMATCH}$$

### 3 Interpretation

#### 3.1 Definitions

Operationally,  $\text{Heap} \sqsubseteq \text{Loc} \times \text{Permission} \times \text{Matrix}$  (a multiset), denoted with a  $\sigma$ .

Define its *interpretation* to be  $\text{Loc} \rightarrow \text{Permission} \times \text{Matrix}$  with  $\star : \text{Heap} \times \text{Heap} \rightarrow \text{Heap}$  as follows:

$$(\varsigma_1 \star \varsigma_2)(l) \equiv \begin{cases} \varsigma_1(l) & \text{if } l \in \text{dom}(\varsigma_1) \wedge l \notin \text{dom}(\varsigma_2) \\ \varsigma_2(l) & \text{if } l \in \text{dom}(\varsigma_2) \wedge l \notin \text{dom}(\varsigma_1) \\ (f_1 + f_2, m) & \text{if } (f_1, m) = \varsigma_1(l) \wedge (f_2, m) = \varsigma_2(l) \wedge f_1 + f_2 \leq 1 \\ \text{undefined} & \text{otherwise} \end{cases}$$

Commutativity and associativity of  $\star$  follows from that of  $+$ .

$\varsigma_1 \star \varsigma_2$  is *defined* if it is for all  $l \in \text{dom}(\varsigma_1) \cup \text{dom}(\varsigma_2)$ .

**Implicitly denote**  $\varsigma \equiv \mathcal{H}[\sigma] \equiv \star_{(l, f, m) \in \sigma} [l \mapsto_f m]$ .

The  $n$ -fold iteration for the *StepsTo* (functional) relation, is also a (functional) relation:

$$\forall n. \mathbf{err} \rightarrow^n \mathbf{err} \quad \langle \sigma, v \rangle \rightarrow^n \langle \sigma, v \rangle \quad \langle \sigma, e \rangle \rightarrow^0 \langle \sigma, e \rangle \quad \langle \sigma, e \rangle \rightarrow^{n+1} ((\langle \sigma, e \rangle \rightarrow) \rightarrow^n)$$

Hence, all bounded iterations end in either an **err**, a heap-and-expression or a heap-and-value.

### 3.2 Interpretation

$$\mathcal{V}_k[\mathbf{unit}] = \{(\emptyset, *)\}$$

$$\mathcal{V}_k[\mathbf{bool}] = \{(\emptyset, true), (\emptyset, false)\}$$

$$\mathcal{V}_k[\mathbf{int}] = \{(\emptyset, n) \mid 2^{-63} \leq n \leq 2^{63} - 1\}$$

$$\mathcal{V}_k[\mathbf{elt}] = \{(\emptyset, f) \mid f \text{ a IEEE Float64 } \}$$

$$\mathcal{V}_k[f \mathbf{mat}] = \{(\{l \mapsto_{2^{-f}} -\}, l)\}$$

$$\mathcal{V}_k[!t] = \{(\emptyset, \mathbf{Many} \ v) \mid (\emptyset, v) \in \mathcal{V}_k[t]\}$$

$$\mathcal{V}_k[\forall fc. t] = \{(\varsigma, \mathbf{fun} \ fc \rightarrow v) \mid \forall f. (\varsigma, (\mathbf{fun} \ fc \rightarrow v)[f]) \in \mathcal{V}_k[t[fc/f]]\}$$

$$\mathcal{V}_k[t_1 \otimes t_2] = \{(\varsigma_1 \star \varsigma_2, (v_1, v_2)) \mid (\varsigma_1, v_1) \in \mathcal{V}_k[t_1] \wedge (\varsigma_2, v_2) \in \mathcal{V}_k[t_2]\}$$

$$\begin{aligned} \mathcal{V}_k[t' \multimap t] = \{(\varsigma_v, v) \mid (v \equiv \mathbf{fun} \ x : t \rightarrow e \vee v \equiv \mathbf{fix}(g, x : t', e : t)) \wedge \\ \forall j \leq k, (\varsigma_{v'}, v') \in \mathcal{V}_j[t']. \varsigma_v \star \varsigma_{v'} \text{ defined} \Rightarrow (\varsigma_v \star \varsigma_{v'}, v \ v') \in \mathcal{C}_j[t]\} \end{aligned}$$

$$\begin{aligned} \mathcal{C}_k[t] = \{(\varsigma_s, e_s) \mid \forall j < k, \sigma_r. \varsigma_s \star \varsigma_r \text{ defined} \Rightarrow \langle \sigma_s + \sigma_r, e_s \rangle \rightarrow^j \mathbf{err} \vee \exists \sigma_f, e_f. \\ \langle \sigma_s + \sigma_r, e_s \rangle \rightarrow^j \langle \sigma_f + \sigma_r, e_f \rangle \wedge (e_f \text{ is a value} \Rightarrow (\varsigma_f \star \varsigma_r, e_f) \in \mathcal{V}_{k-j}[t])\} \end{aligned}$$

$$\mathcal{I}_k[\cdot]\theta = \{\emptyset\}$$

$$\mathcal{I}_k[\Delta, x : t]\theta = \{\delta[x \mapsto v_x] \mid \delta \in \mathcal{I}_k[\Delta]\theta \wedge (\emptyset, v_x) \in \mathcal{V}_k[\theta(t)]\}$$

$$\mathcal{L}_k[\cdot]\theta = \{(\emptyset, [])\}$$

$$\mathcal{L}_k[\Gamma, x : t]\theta = \{(\varsigma \star \varsigma_x, \gamma[x \mapsto v_x]) \mid (\varsigma, \gamma) \in \mathcal{L}_k[\Gamma]\theta \wedge (\varsigma_x, v_x) \in \mathcal{V}_k[\theta(t)]\}$$

$$\varsigma \equiv \mathcal{H}[\sigma] \equiv \star_{(l, f, m) \in \sigma} [l \mapsto_f m]$$

$$\begin{aligned} {}_k[\Theta; \Delta; \Gamma \vdash e : t] = \forall \theta, \delta, \gamma, \sigma. \text{dom}(\Theta) = \text{dom}(\theta) \wedge (\varsigma, \gamma) \in \mathcal{L}_k[\Gamma]\theta \wedge \delta \in \mathcal{I}_k[\Delta]\theta \Rightarrow \\ (\varsigma, \gamma(\delta(e))) \in \mathcal{C}_k[\theta(t)] \end{aligned}$$

## 4 Proofs

### 4.1 Lemmas

**4.1.1**  $\forall \sigma_s, \sigma_r, e. \varsigma_s \star \varsigma_r \text{ defined} \Rightarrow \forall n. \langle \sigma_s, e \rangle \rightarrow^n \langle \sigma_s + \sigma_r, e \rangle \rightarrow^n$

SUFFICES: By induction on  $n$ , consider only the cases  $\langle \sigma_s, e \rangle \rightarrow \langle \sigma_f, e_f \rangle$  where  $\sigma_s \neq \sigma_f$ .

PROOF SKETCH: Only `OP_FREE`, `MATRIX`, `SHARE`, `UNSHARE_EQ`, `GEMM_MATCH` change the heap: the rest are either parametric in the heap or step to an **err**.

PROVE:  $\langle \sigma_s + \sigma_r, e \rangle \rightarrow \langle \sigma_f + \sigma_r, e_f \rangle$ .

$\langle 1 \rangle 1$ . CASE: `OP_FREE`,  $\sigma_s \equiv \sigma' + \{l \mapsto_1 m\}$ ,  $\sigma_f = \sigma'$ .

PROOF: Instantiate `OP_FREE` with  $(\sigma' + \sigma_r) + \{l \mapsto_1 m\}$ ,  
valid because  $l \notin \text{dom}(\varsigma_r)$  by  $\varsigma' \star [l \mapsto_1 m] \star \varsigma_r$  defined (assumption).

$\langle 1 \rangle 2$ . CASE: `OP_MATRIX`

PROOF: Rule has no requirements on  $\sigma_s$  so will also work with  $\sigma_s + \sigma_r$ .

$\langle 1 \rangle 3$ . CASE: `OP_SHARE`,  $\sigma_s \equiv \sigma' + \{l \mapsto_f m\}$ ,  $\sigma_f = \sigma' + \{l \mapsto_{\frac{1}{2}.f} m\} + \{l \mapsto_{\frac{1}{2}.f} m\}$ .

PROOF: Union-ing  $\sigma_r$  does not remove  $l \mapsto_f m$ , so that can be split out of  $\sigma_s + \sigma_r$  as before.

$\langle 1 \rangle 4$ . CASE: `OP_UNSHARE_EQ`,  $\sigma_s \equiv \sigma' + \{l \mapsto_{\frac{1}{2}.f} m\} + \{l \mapsto_{\frac{1}{2}.f} m\}$ ,  $\sigma_f = \sigma' + \{l \mapsto_f m\}$ .

$\langle 2 \rangle 1$ . Union-ing  $\sigma_r$  does not remove  $l \mapsto_{\frac{1}{2}.f} m$ , so that can still be split out of  $\sigma_s + \sigma_r$ .

$\langle 2 \rangle 2$ . There may also be other valid splits introduced by  $\sigma_r$ .

$\langle 2 \rangle 3$ . However, by assumption of  $\varsigma_s \star \varsigma_r$  defined, any splitting of  $\sigma_s + \sigma_r$  will satisfy  $f \leq 1$ .

$\langle 1 \rangle 5$ . CASE: `OP_GEMM_MATCH`

$\langle 2 \rangle 1$ . By assumption of  $\varsigma_s \star \varsigma_r$  defined, either  $l_1$  (or  $l_2$ , or both) are not in  $\sigma_r$ , or they are and the matrix values they point to are the same.

$\langle 2 \rangle 2$ . The permissions (of  $l_1$  and/or  $l_2$ ) may differ, but `OP_GEMM_MATCH` universally quantifies over them and leaves them unchanged, so they are irrelevant.

$\langle 2 \rangle 3$ . Only the pointed to matrix value at  $l_3$  changes.

$\langle 2 \rangle 4$ . SUFFICES:  $l_3 \notin \pi_1[\sigma_r]$ .

$\langle 2 \rangle 5$ . By assumption of  $\varsigma_s \star \varsigma_r$  defined,  $l_3 \notin \text{dom}(\varsigma_r)$ .

$\langle 2 \rangle 6$ . Hence  $l_3 \notin \pi_1[\sigma_r]$ .

**4.1.2**  $\forall k, t. \mathcal{V}_k[t] \subseteq \mathcal{C}_k[t]$

Follows from definition of  $\mathcal{C}_k[t]$ ,  $\rightarrow^j$  ( $\forall n. \langle \sigma, v \rangle \rightarrow^n \langle \sigma, v \rangle$ ) for arbitrary  $j \leq k$  and 4.1.1.

**4.1.3**  $\forall \delta, \gamma, v. \delta(\gamma(v))$  is a value.

By construction,  $\delta$  and  $\gamma$  only map variables to values, and values are closed under substitution.

**4.1.4**  $\forall k, \sigma, \sigma', e, e', t. (\varsigma', e') \in \mathcal{C}_k[t] \wedge \langle \sigma, e \rangle \rightarrow \langle \sigma', e' \rangle \Rightarrow (\varsigma, e) \in \mathcal{C}_{k+1}[t]$

ASSUME: arbitrary  $j < k + 1$ , and  $\sigma_r$  such that  $\varsigma \star \varsigma_r$  defined.

$\langle 1 \rangle 1$ . CASE:  $j = 0$ . Clearly  $\sigma_f = \sigma_s + \sigma_r$  and  $e' = e$ .

Remains to show that if  $e$  is a value then  $(\varsigma_s \star \varsigma_r, e) \in \mathcal{V}_k[t]$ .

This is true vacuously, because by assumption,  $e$  is not a value.

$\langle 1 \rangle 2$ . CASE:  $j \geq 1$ . We have  $\langle \sigma, e \rangle \rightarrow^j = \langle \sigma', e' \rangle \rightarrow^{j-1}$ .

Instantiate  $(\varsigma', e') \in \mathcal{C}_k[t]$ , with  $j - 1 < k$  and  $\sigma_r$  to conclude the required conditions.

**4.1.5**  $j \leq k \Rightarrow \_k[\cdot] \subseteq \_j[\cdot]$

Lemma 4.1.4 is the inductive step for this lemma for the  $\mathcal{C}[\cdot]$  case.

Need to prove for  $\mathcal{V}[\cdot]$ , by induction on  $t$  and then index.

SUFFICES: Consider only  $t \multimap t'$  case, rest use  $k$  directly on structure of type.

ASSUME: Arbitrary  $j \leq k$  and  $(\varsigma_{v'}, v') \in \mathcal{V}_k[t \multimap t']$ .

PROVE:  $(\varsigma_{v'}, v') \in \mathcal{V}_j[t \multimap t']$ .

$\langle 1 \rangle 1$ .  $v'$  is of the correct syntactic form (lambda or fixpoint) by assumption.

$\langle 1 \rangle 2$ . ASSUME: arbitrary  $j' \leq j$  and  $(\varsigma_v, v) \in \mathcal{V}_{j'}[t]$  such that  $\varsigma_{v'} \star \varsigma_v$  is defined.

$\langle 1 \rangle 3$ . SUFFICES: to show  $(\varsigma_{v'} \star \varsigma_v, v'v) \in \mathcal{C}_{j'}[t']$ .

$\langle 1 \rangle 4$ . This is true by instantiating  $(\varsigma_{v'}, v') \in \mathcal{V}_k[t \multimap t']$  with  $j' \leq k$  and  $(\varsigma_v, v) \in \mathcal{V}_{j'}[t]$ .

**4.1.6**  $\forall \Delta, \Gamma, t, k, \theta, \delta, \gamma. \delta \in \mathcal{I}_k[\Delta]\theta \wedge \gamma \in \pi_2[\mathcal{L}_k[\Gamma]\theta] \Rightarrow \text{dom}(\Delta) = \text{dom}(\delta) \text{ and } \text{dom}(\Gamma) = \text{dom}(\gamma)$

PROOF: By induction on  $\Delta$  and  $\Gamma$ .

**4.1.7**  $\forall k, \Gamma, \Gamma', \theta, \sigma_+, \gamma_+. (\varsigma_+, \gamma) \in \mathcal{L}_k[\Gamma, \Gamma']\theta \wedge \Gamma, \Gamma' \text{ disjoint} \Rightarrow$   
 $\exists \sigma, \gamma, \sigma', \gamma'. \sigma_+ = \sigma + \sigma' \wedge \gamma, \gamma' \text{ disjoint} \wedge \gamma_+ = \gamma \cup \gamma'$   
 $\wedge (\varsigma, \gamma) \in \mathcal{L}_k[\Gamma] \wedge (\varsigma', \gamma') \in \mathcal{L}_k[\Gamma']$

PROOF: By induction on  $\Gamma'$ .

## 4.2 Soundness

$$\forall \Theta, \Delta, \Gamma, e, t. \Theta; \Delta; \Gamma \vdash e : t \Rightarrow \forall k. {}_k\llbracket \Theta; \Delta; \Gamma \vdash e : t \rrbracket$$

PROOF SKETCH: Induction over the typing judgements.

ASSUME: 1. Arbitrary  $\Theta, \Delta, \Gamma, e, t$  such that  $\Theta; \Delta; \Gamma \vdash e : t$ .

2. Arbitrary  $k, \theta, \delta, \gamma, \sigma$  such that:

a.  $\text{dom}(\Theta) = \text{dom}(\theta)$

b.  $(\varsigma, \gamma) \in \mathcal{L}_k\llbracket \Gamma \rrbracket \theta$

c.  $\delta \in \mathcal{I}_k\llbracket \Delta \rrbracket \theta$ .

3. W.l.o.g., all variables are distinct,

hence  $\text{dom}(\Delta)$  and  $\text{dom}(\Gamma)$  are disjoint

so  $\gamma \circ \delta = \delta \circ \gamma$  (as substitutions defined recursively over expressions).

PROVE:  $(\varsigma, \gamma(\delta(e))) \in \mathcal{C}_k\llbracket \theta(t) \rrbracket$ .

ASSUME: Arbitrary  $j < k$  and  $\sigma_r$ , such that  $\varsigma \star \varsigma_r$  defined.

SUFFICES:  $\langle \sigma + \sigma_r, e \rangle \rightarrow^j \mathbf{err} \vee \exists \sigma_f, e_f. \langle \sigma + \sigma_r, e \rangle \rightarrow^j \langle \sigma_f + \sigma_r, e_f \rangle$

$\wedge (e_f \text{ is a value} \Rightarrow (\varsigma_f \star \varsigma_r, e_f) \in \mathcal{V}_{k-j}\llbracket t \rrbracket)$ .

SUFFICES: By 4.1.1, to show  $\langle \sigma, e \rangle \rightarrow^j \mathbf{err} \vee \exists \sigma_f, e_f. \langle \sigma, e \rangle \rightarrow^j \langle \sigma_f, e_f \rangle$

$\wedge (e_f \text{ is a value} \Rightarrow (\varsigma_f, e_f) \in \mathcal{V}_{k-j}\llbracket t \rrbracket)$

$\langle 1 \rangle 1$ . CASE: **TY-LET**.

$\langle 2 \rangle 1$ . By induction,

1.  $\forall k. {}_k\llbracket \Theta; \Delta; \Gamma \vdash e : t \rrbracket$

2.  $\forall k. {}_k\llbracket \Theta; \Delta; \Gamma', x : t \vdash e' : t' \rrbracket$ .

$\langle 2 \rangle 2$ . By 2b, 3 and 4.1.7, we know there exists the following (for all  $k$ ):

1.  $(\varsigma_e, \gamma_e) \in \mathcal{L}_k\llbracket \Gamma \rrbracket$

2.  $\gamma = \gamma_e \cup \gamma_{e'}$

3.  $\sigma = \sigma_e + \sigma_{e'}$ .

$\langle 2 \rangle 3$ . So, using  $k, \theta, \delta, \gamma_e, \sigma_e$ , we have  $(\varsigma_e, \gamma_e(\delta(e))) \in \mathcal{C}_k\llbracket \theta(t) \rrbracket$ .

$\langle 2 \rangle 4$ . By  $\langle 2 \rangle 2$  ( $\gamma = \gamma_e \cup \gamma_{e'}$ ), have  $(\varsigma_e, \gamma(\delta(e))) \in \mathcal{C}_k\llbracket \theta(t) \rrbracket$ .

$\langle 2 \rangle 5$ . By definition of  $\mathcal{C}_k\llbracket \cdot \rrbracket$  and  $\langle 2 \rangle 2$ , we instantiate with  $j$  and  $\sigma_r = \sigma_{e'}$  to conclude that  $\langle \varsigma, \gamma(\delta(e)) \rangle$  either takes  $j$  steps to **err** or another heap-and-expression  $\langle \sigma_f, \gamma(\delta(e_f)) \rangle$ .

$\langle 2 \rangle 6$ . CASE:  $j$  steps to **err**

By **OP\_CONTEXT\_ERR**, the whole expression reduces to **err** in  $j < k$  steps.

$\langle 2 \rangle 7$ . CASE:  $j$  steps to another heap-and-expression.

If it is not a value, then **OP\_CONTEXT** runs  $j$  times and we are done.

$\langle 2 \rangle 8$ . If it is, then  $\exists i \leq j. (\varsigma_f, v_1) \in \mathcal{V}_{k-i}\llbracket \theta(t_1) \rrbracket \subseteq \mathcal{V}_{k-j}\llbracket \theta(t_1) \rrbracket$  by 4.1.3 and 4.1.5.

So, **OP\_CONTEXT** runs  $i$  times, and then we have the following.

SUFFICES:  $(\varsigma_f \star \varsigma_{e'}, \mathbf{let } x = v \mathbf{ in } \gamma(\delta(e'))) \in \mathcal{C}_{k-i}\llbracket \theta(t') \rrbracket$  by 4.1.4  $i$  times.

SUFFICES:  $(\varsigma_f \star \varsigma_{e'}, \gamma(\delta(e'))[x/v]) \in \mathcal{C}_{k-i-1}\llbracket \theta(t') \rrbracket$  by 4.1.4.

$\langle 2 \rangle 9$ . By 4.1.5,  $(\varsigma_{e'}, \gamma_{e'}[x \mapsto v]) \in \mathcal{L}_k\llbracket \Gamma', x : t \rrbracket \theta \subseteq \mathcal{L}_{k-i-1}\llbracket \Gamma', x : t \rrbracket \theta$ .

$\langle 2 \rangle 10$ . Instantiate 2 of step  $\langle 2 \rangle 1$  with  $k - i - 1, \theta, \delta, \gamma_{e'}[x \mapsto v], \sigma_{e'}$  to conclude

$$(\varsigma_{e'}, \gamma_{e'}[x \mapsto v](\delta(e'))) \in \mathcal{C}_{k-i-1}[\![\theta(t')]\!].$$

$\langle 2 \rangle 11$ . By 3, we have  $\gamma(\delta(e'))[x/v] = \gamma_{e'}[x \mapsto v](\delta(e'))$  and by 4.1.1 we conclude  $(\varsigma_f \star \varsigma_{e'}, \gamma(\delta(e'))[x/v]) \in \mathcal{C}_{k-i-1}[\![\theta(t')]\!]$

$\langle 1 \rangle 2$ . CASE: `TY_PAIR_ELIM`.

PROOF SKETCH: Similar to `TY_LET`, but with the following key differences.

$\langle 2 \rangle 1$ . When  $(\varsigma_f, v) \in \mathcal{V}_{k-i}[\![\theta(t_1) \otimes \theta(t_2)]\!]$ , we have  $v = (v_1, v_2)$ .

$\langle 2 \rangle 2$ . SUFFICES:  $(\varsigma_{e'}, \gamma(\delta(e'))) \in \mathcal{C}_{k-i-1}[\![\theta(t')]\!]$  by 4.1.4  $i + 1$  times.

$\langle 2 \rangle 3$ . By 4.1.5,  $(\varsigma_{e'}, \gamma_{e'}[a \mapsto v_1, b \mapsto v_2]) \in \mathcal{L}_k[\![\Gamma', a : t_1, b : t_2]\!]\theta \subseteq \mathcal{L}_{k-i-1}[\![\Gamma', a : t_1, b : t_2]\!]\theta$ .

$\langle 2 \rangle 4$ . Instantiate  $_{k-i-1}[\![\Theta; \Delta; \Gamma', a : t_1, b : t_2 \vdash e' : t']\!]$  with  $\theta, \delta, \gamma_{e'}[a \mapsto v_1, b \mapsto v_2], \sigma_{e'}$ .

$\langle 2 \rangle 5$ . By 3 (for  $\gamma = \gamma_e \cup \gamma_{e'}$  and  $a, b$ ), conclude  $(\varsigma_{e'}, \gamma(\delta(e'[a/v_1][b/v_2]))) \in \mathcal{C}_{k-i-1}[\![\theta(t')]\!]$ .

$\langle 1 \rangle 3$ . CASE: `TY_BANG_ELIM`.

PROOF SKETCH: Similar to `TY_LET`, but with the following key differences.

$\langle 2 \rangle 1$ . When  $(\varsigma_f, v) \in \mathcal{V}_{k-i}[\![\theta(!t)]\!]$ , since  $\mathcal{V}_{k-i}[\![\theta(!t)]\!] = \mathcal{V}_{k-i}[\![! \theta(t)]\!]$ , we have  $\varsigma_f = \emptyset$  and  $v = \mathbf{Many} \ v'$  for some  $(\emptyset, v') \in \mathcal{V}_{k-i}[\![\theta(t)]\!]$ .

$\langle 2 \rangle 2$ . SUFFICES:  $(\varsigma_{e'}, \mathbf{let} \ \mathbf{Many} \ x = \mathbf{Many} \ v' \ \mathbf{in} \ \gamma(\delta(e'))) \in \mathcal{C}_{k-i}[\![\theta(t)]\!]$ .

$\langle 2 \rangle 3$ . SUFFICES:  $(\varsigma_{e'}, \gamma(\delta(e'))[x/v]) \in \mathcal{C}_{k-i-1}[\![\theta(t)]\!]$  by 4.1.4  $i + 1$  times.

$\langle 2 \rangle 4$ . Instantiate  $_{k-i-1}[\![\Theta; \Delta, x : t, \Gamma' \vdash e' : t']\!]$  with  $\theta, \delta_{e'} = \delta[x \mapsto v'], \gamma_{e'}, \sigma_{e'}$ .

$\langle 2 \rangle 5$ . By 3,  $(\varsigma_{e'}, \gamma(\delta(e'))[x/v]) \in \mathcal{C}_{k-i-1}[\![\theta(t)]\!]$ .

$\langle 1 \rangle 4$ . CASE: `TY_UNIT_ELIM`.

PROOF SKETCH: Similar to `TY_LET`, but with the following key differences.

$\langle 2 \rangle 1$ . When  $(\varsigma_f, v) \in \mathcal{V}_{k-i}[\![\mathbf{unit}]\!]$ , we have  $\varsigma_f = \emptyset$  and  $v = ()$ .

$\langle 2 \rangle 2$ . SUFFICES:  $(\varsigma_{e'}, \gamma(\delta(e'))) \in \mathcal{C}_{k-i-1}[\![\theta(t')]\!]$  by 4.1.4  $i + 1$  times.

$\langle 2 \rangle 3$ . By 4.1.5,  $(\varsigma_{e'}, \gamma_{e'}) \in \mathcal{L}_k[\![\Gamma']\!]\theta \subseteq \mathcal{L}_{k-i-1}[\![\Gamma']\!]\theta$ .

$\langle 2 \rangle 4$ . Instantiate  $_{k-i-1}[\![\Theta; \Delta; \Gamma' \vdash e' : t']\!]$  with  $\theta, \delta, \gamma_{e'}, \sigma_{e'}$ .

$\langle 2 \rangle 5$ . By 3  $(\varsigma_{e'}, \gamma(\delta(e'))) \in \mathcal{C}_{k-i-1}[\![\theta(t')]\!]$ .

$\langle 1 \rangle 5$ . CASE: `TY_BOOL_ELIM`.

PROOF SKETCH: Similar to `TY_UNIT_ELIM` but with `OP_IF_{-}\{\mathbf{TRUE}, \mathbf{FALSE}\}`,  $\varsigma_f = \emptyset$  and  $v = \mathbf{Many} \ \mathbf{true}$  or  $v = \mathbf{Many} \ \mathbf{false}$ .

$\langle 1 \rangle 6$ . CASE: `TY_BANG_INTRO`.

$\langle 2 \rangle 1$ . We have,  $e = v$  for some value  $v \neq l \cdot f$ ,  $\Gamma = \emptyset$  and so  $\forall k. \ _k[\![\Theta; \Delta; \cdot \vdash v : t]\!]$  by induction.

$\langle 2 \rangle 2$ . SUFFICES:  $(\emptyset, \mathbf{Many} \ \delta(v)) \in \mathcal{C}_k[\![! \theta(t)]\!]$  by 2b ( $\varsigma = \emptyset, \gamma = []$ ).

$\langle 2 \rangle 3$ . Instantiate  $_k[\![\Theta; \Delta; \cdot \vdash v : t]\!]$  with  $\theta, \delta, \gamma = [], \sigma = \emptyset$  to obtain  $(\emptyset, \delta(v)) \in \mathcal{C}_k[\![\theta(t)]\!]$ .



- ⟨2⟩4. Instantiate  $(\emptyset, \delta(v)) \in \mathcal{C}_k[\![\theta(t)]\!]$  with  $j = 0$ ,  $\sigma_r = \emptyset$  and 4.1.3 ( $\delta(v)$  is a value), to conclude  $(\emptyset, \delta(v)) \in \mathcal{V}_k[\![\theta(t)]\!]$ .
- ⟨2⟩5. By definition of  $\mathcal{V}_k[\![\theta(t)]\!]$ , 4.1.3 and 4.1.2 we have  $(\emptyset, \mathbf{Many} \delta(v)) \in \mathcal{C}_k[\![\theta(t)]\!]$ .
- ⟨1⟩7. CASE: `TY_PAIR_INTRO`.
- ⟨2⟩1. By 2b, 3 and 4.1.7, we know there exists the following (for all  $k$ ):
1.  $(\varsigma_1, \gamma_1) \in \mathcal{L}_k[\![\Gamma_1]\!]$
  2.  $(\varsigma_2, \gamma_2) \in \mathcal{L}_k[\![\Gamma_2]\!]$
  3.  $\gamma = \gamma_1 \cup \gamma_2$
  4.  $\sigma = \sigma_1 + \sigma_2$ .
- ⟨2⟩2. By induction,
1.  $\forall k. {}_k[\![\Theta; \Delta; \Gamma_1 \vdash e_1 : t_1]\!]$
  2.  $\forall k. {}_k[\![\Theta; \Delta; \Gamma_2 \vdash e_2 : t_2]\!]$ .
- ⟨2⟩3. Instantiate the first with  $k, \theta, \delta, \gamma_1, \sigma_1$ .
- ⟨2⟩4. By that and ⟨2⟩1,  $(\varsigma_1, \gamma_1(\delta(e_1))) = (\varsigma_1, \gamma(\delta(e_1))) \in \mathcal{C}_k[\![\theta(t)]\!]$ .
- ⟨2⟩5. So,  $\langle \sigma_1 + \sigma_2, \gamma_1(\delta(e_1)) \rangle$  either takes  $j$  steps to **err** or a heap-and-expression  $\langle \sigma_{1f}, e_{1f} \rangle$ .
- ⟨2⟩6. CASE:  $j$  steps to **err**  
By `OP_CONTEXT_ERR`, the whole expression reduces to **err** in  $j < k$  steps.
- ⟨2⟩7. CASE:  $j$  steps to another heap-and-expression.  
If it is not a value, then `OP_CONTEXT` runs  $j$  times and we are done.
- ⟨2⟩8. If it is, then  $\exists i_1 \leq j. (\varsigma_{1f}, v_1) \in \mathcal{V}_{k-i_1}[\![\theta(t_1)]\!] \subseteq \mathcal{V}_{k-j}[\![\theta(t_1)]\!]$  by 4.1.3 and 4.1.5.  
So, `OP_CONTEXT` runs  $i_1$  times, and then we have the following.  
SUFFICES: By 4.1.4,  $(\varsigma_{1f} \star \varsigma_2, (v_1, e_2)) \in \mathcal{C}_{k-i_1}[\![\theta(t_1 \otimes t_2)]\!]$ .
- ⟨2⟩9. Instantiate the second IH with  $k, \theta, \delta, \gamma_2, \sigma_2$ .
- ⟨2⟩10. So,  $\langle \sigma_{1f} \star \sigma_2, \gamma_2(\delta(e_2)) \rangle$  either takes  $j$  steps to **err** or a heap-and-expression  $\langle \sigma_{2f}, e_{2f} \rangle$ .
- ⟨2⟩11. CASE:  $j$  steps to **err**  
By `OP_CONTEXT_ERR`, the whole expression reduces to **err** in  $j < k$  steps.
- ⟨2⟩12. CASE:  $j$  steps to another heap-and-expression.  
If it is not a value, then `OP_CONTEXT` runs  $j$  times and we are done.
- ⟨2⟩13. If it is, then  $\exists i_2 \leq j. (\varsigma_{2f}, v_2) \in \mathcal{V}_{k-i_2}[\![\theta(t_2)]\!] \subseteq \mathcal{V}_{k-j}[\![\theta(t_2)]\!]$  by 4.1.3 and 4.1.5.  
So, `OP_CONTEXT` runs  $i_2$  times, and then we have the following.  
SUFFICES: By 4.1.4,  $(\varsigma_{1f} \star \varsigma_{2f}, (v_1, v_2)) \in \mathcal{V}_{k-i_1-i_2}[\![\theta(t_1) \otimes \theta(t_2)]\!]$ .
- ⟨2⟩14. By 4.1.5 and  $k - i_1 - i_2 \leq k - i_1, k - i_2$ , have  
 $(\varsigma_{1f}, v_1) \in \mathcal{V}_{k-i_1}[\![\theta(t_1)]\!] \subseteq \mathcal{V}_{k-i_1-i_2}[\![\theta(t_1)]\!]$  and  
 $(\varsigma_{2f}, v_2) \in \mathcal{V}_{k-i_2}[\![\theta(t_2)]\!] \subseteq \mathcal{V}_{k-i_1-i_2}[\![\theta(t_2)]\!]$  as needed.
- ⟨1⟩8. CASE: `TY_LAMBDA`.  
SUFFICES: By 4.1.2, to show  $(\varsigma, \gamma(\delta(\mathbf{fun} x : t \rightarrow e))) \in \mathcal{V}_k[\![\theta(t \multimap t')]\!]$ .  
ASSUME: Arbitrary  $j \leq k$ ,  $(\varsigma_v, v) \in \mathcal{V}_j[\![\theta(t)]\!]$  such that  $\varsigma \star \varsigma_v$  is defined.  
SUFFICES:  $(\varsigma \star \varsigma_v, \gamma(\delta(\mathbf{fun} x : t \rightarrow e)) v) \in \mathcal{C}_j[\![\theta(t')]\!]$ .  
SUFFICES:  $(\varsigma \star \varsigma_v, \gamma(\delta(e))[x/v]) \in \mathcal{C}_{j-1}[\![\theta(t')]\!]$  by 4.1.4.

- ⟨2⟩1. By induction,  $\forall k. {}_k\llbracket \Theta; \Delta; \Gamma, x : t \vdash e \rrbracket$ .
- ⟨2⟩2. Instantiate it  $j - 1, \theta, \gamma[x \mapsto v], \sigma + \sigma_v$ .
- ⟨2⟩3. Hence,  $(\varsigma \star \varsigma_v, \gamma[x \mapsto v](\delta(e))) \in \mathcal{C}_{j-1}\llbracket \theta(t) \rrbracket$ .
- ⟨2⟩4. By 3,  $\gamma[x \mapsto v](\delta(e)) = \gamma(\delta(e))[x/v]$ , we are done.
- ⟨1⟩9. CASE: `TY_APP`.
- ⟨2⟩1. By 2b, 3 and 4.1.7, we know there exists the following (for all  $k$ ):
1.  $(\varsigma_e, \gamma_e) \in \mathcal{L}_k\llbracket \Gamma_e \rrbracket$
  2.  $(\varsigma_{e'}, \gamma_{e'}) \in \mathcal{L}_k\llbracket \Gamma_{e'} \rrbracket$
  3.  $\gamma = \gamma_e \cup \gamma_{e'}$
  4.  $\sigma = \sigma_e + \sigma_{e'}$ .
- ⟨2⟩2. By induction,
1.  $\forall k. {}_k\llbracket \Theta; \Delta; \Gamma \vdash e : t' \multimap t \rrbracket$
  2.  $\forall k. {}_k\llbracket \Theta; \Delta; \Gamma' \vdash e' : t' \rrbracket$ .
- ⟨2⟩3. Instantiate the first with  $\theta, k, \delta, \gamma_e, \sigma_e$  to conclude  $(\varsigma_e, \gamma_e(\delta(e))) \in \mathcal{C}_k\llbracket \theta(t') \multimap \theta(t) \rrbracket$ .
- ⟨2⟩4. Instantiate *this* with  $j$  and  $\sigma_{e'}$  and use ⟨2⟩1 to conclude  $\langle \sigma_e + \sigma_{e'}, \gamma(\delta(e)) \rangle$  either takes  $j$  steps to **err** or a heap-and-expression  $\langle \sigma_f + \sigma_{e'}, e_f \rangle$ .
- ⟨2⟩5. CASE:  $j$  steps to **err**  
By `OP_CONTEXT_ERR`, the whole expression reduces to **err** in  $j < k$  steps.
- ⟨2⟩6. CASE:  $j$  steps to another heap-and-expression.  
If it is not a value, then `OP_CONTEXT` runs  $j$  times and we are done.
- ⟨2⟩7. If it is, then  $\exists i_e \leq j. (\varsigma_f, v_e) \in \mathcal{V}_{k-i_e}\llbracket \theta(t') \multimap \theta(t) \rrbracket \subseteq \mathcal{V}_{k-j}\llbracket \dots \rrbracket$  by 4.1.3 and 4.1.5.  
So, `OP_CONTEXT` runs  $i_e$  times, and then we have the following.  
SUFFICES: By 4.1.4  $i_e$  times,  $(\varsigma_f \star \varsigma_{e'}, v_e e') \in \mathcal{C}_{k-i_e}\llbracket \theta(t') \rrbracket$ .
- ⟨2⟩8. By 4.1.5,  $(\varsigma_{e'}, \gamma_{e'} \in \mathcal{L}_k\llbracket \Gamma' \rrbracket \theta \subseteq \mathcal{L}_{k-i_e}\llbracket \Gamma' \rrbracket \theta$ .
- ⟨2⟩9. So, instantiate the second IH with  $k - i_e, \theta, \delta, \gamma_{e'}, \sigma_{e'}$  to conclude  $(\varsigma_{e'}, \gamma_{e'}(\delta(e'))) \in \mathcal{C}_{k-i_e}\llbracket \theta(t') \rrbracket$ .
- ⟨2⟩10. Instantiate *this* with  $j - i_e$  and  $\sigma_f$  to conclude  $\langle \sigma_f + \sigma_{e'}, \gamma_{e'}(\delta(e')) \rangle$  either takes  $j - i_e$  steps to **err** or  $\langle \sigma_f + \sigma_{f'}, e_{f'} \rangle$ .
- ⟨2⟩11. CASE:  $j - i_e$  steps to **err**  
By `OP_CONTEXT_ERR`, the whole expression reduces to **err** in  $j - i_e < k - i_e$  steps.
- ⟨2⟩12. CASE:  $j - i_e$  steps to another heap-and-expression.  
If it is not a value, then `OP_CONTEXT` runs  $j - i_e$  times and we are done.
- ⟨2⟩13. If it is, then  $\exists i_{e'} \leq j - i_e. (\varsigma_{f'}, v_{e'}) \in \mathcal{V}_{k-i_e-i_{e'}}\llbracket \theta(t') \rrbracket$  by 4.1.3.  
So, `OP_CONTEXT` runs  $i_{e'}$  times, and then we have the following.  
SUFFICES: By 4.1.4  $i_{e'}$  times,  $(\varsigma_f \star \varsigma_{f'}, v_e v_{e'}) \in \mathcal{C}_{k-i_e-i_{e'}}\llbracket \theta(t') \rrbracket$ .
- ⟨2⟩14. Instantiate  $(\varsigma_f, v_e) \in \mathcal{V}_{k-i_e}\llbracket \theta(t') \multimap \theta(t) \rrbracket$  with  $k - i_e - i_{e'} \leq k - i_e$  and  $(\varsigma_{v'}, v_{e'}) \in \mathcal{V}_{k-i_e-i_{e'}}\llbracket \theta(t') \rrbracket$ , to conclude  $(\varsigma_f \star \varsigma_{f'}, v_e v_{e'}) \in \mathcal{C}_{k-i_e-i_{e'}}\llbracket \theta(t) \rrbracket$  as needed.

⟨1⟩10. CASE: TY\_GEN.

PROVE:  $(\sigma, \gamma(\delta(\mathbf{fun} \, fc \rightarrow e))) \in \mathcal{C}_k[\![\theta(\forall \, fc. t)]\!]$ .

⟨1⟩11. CASE: TY\_SPC.

PROVE:  $(\sigma, \gamma(\delta(e[f]))) \in \mathcal{C}_k[\![\theta(t[f c/f])]\!]$ .

⟨1⟩12. CASE: TY\_FIX.

PROVE:  $(\sigma, \gamma(\delta(\mathbf{fix}(g, x : t, e : t')))) \in \mathcal{C}_k[\![\theta(! (t \multimap t'))]\!]$ .

SUFFICES: to show  $\dots \in \mathcal{V}_k[\![\theta(t) \multimap \theta(t')]\!]$ , by 4.1.2.

⟨2⟩1. ASSUME: Arbitrary  $j < k$  and  $(\sigma, v) \in \mathcal{V}_j[\![\theta(t)]\!]$ .

⟨2⟩2. SUFFICES:  $(\sigma, \mathit{letManyG} \, g \, v) \in \mathcal{C}_j[\![\theta(t')]\!]$ .

⟨2⟩3. LET:  $e_1 = e[g/\mathbf{fun} \, x : t \rightarrow \mathit{letManyG} \, g \, x]$ .

⟨2⟩4. SUFFICES: by 4.1.4,  $(\sigma, (\mathbf{fun} \, x : t \rightarrow e_1) \, v) \in \mathcal{C}_{j-1}[\![\theta(t')]\!]$ .

⟨2⟩5. SUFFICES: by 4.1.4,  $(\sigma, e_1[x/v]) \in \mathcal{C}_{j-2}[\![\theta(t')]\!]$ .

⟨2⟩6. By induction, we have  $\llbracket \Theta; \Delta, g : t \multimap t'; x : t \vdash e : t' \rrbracket$ .

⟨2⟩7. Instantiate this with  $\theta, j-2, \delta[g \mapsto \mathbf{fun} \, x : t \rightarrow e_1], \gamma = [x \mapsto v], \sigma$  (???).

PROVE:  $(\sigma, \mathbf{fun} \, x : t \rightarrow e_1) \in \mathcal{V}_{j-2}[\![\theta(t) \multimap \theta(t')]\!]$ .

⟨3⟩1. SUFFICES: by 4.1.4,  $(\sigma', e_1[x/v']) \in \mathcal{C}_{j-2}[\![\theta(t')]\!]$  for arbitrary  $(\sigma', v') \in \mathcal{V}_{j-2}[\![\theta(t)]\!]$ .

⟨3⟩2. We can again use the induction hypothesis  $\llbracket \Theta; \Delta, g : t \multimap t'; x : t \vdash e : t' \rrbracket$ .

⟨3⟩3. But since it's true for  $\mathcal{C}_0[\![\cdot]\!]$  (base case), it's true by induction ???

⟨2⟩8. Lastly, we show  $\delta(\gamma(e)) = e_1[x/v]$ , which follows by their definitions, to conclude  $(\sigma, e_1[x/v]) \in \mathcal{C}_{j-2}[\![\theta(t')]\!]$ .

⟨1⟩13. CASE: TY\_VAR\_LIN.

PROVE:  $(\sigma, \gamma(\delta(x))) \in \mathcal{C}_k[\![\theta(t)]\!]$ .

⟨2⟩1.  $\Gamma = \{x : t\}$  by assumption of TY\_VAR\_LIN.

⟨2⟩2. SUFFICES:  $(\sigma, \gamma(x)) \in \mathcal{C}_k[\![\theta(t)]\!]$  by 3.

⟨2⟩3. By 2b, there exist  $(\sigma_x, v_x) \in \mathcal{V}_k[\![\theta(t)]\!]$ , such that  $\sigma = \sigma_x$  and  $\gamma = [x \mapsto v_x]$ .

⟨2⟩4. Hence,  $(\sigma_x, v_x) \in \mathcal{C}_k[\![\theta(t)]\!]$ , by 4.1.2.

⟨1⟩14. CASE: TY\_VAR.

PROVE:  $(\sigma, \gamma(\delta(x))) \in \mathcal{C}_k[\![\theta(t)]\!]$ .

⟨2⟩1.  $x : t \in \Delta$  and  $\Gamma = \emptyset$  by assumption of TY\_VAR.

⟨2⟩2. SUFFICES:  $(\emptyset, \delta(x)) \in \mathcal{C}_k[\![\theta(t)]\!]$  by 3 and 2b.

⟨2⟩3. By 2c, there exists  $v_x$  such that  $(\emptyset, v_x) \in \mathcal{V}_k[\![\theta(t)]\!]$ .

$\langle 2 \rangle 4$ . Hence,  $(\emptyset, v_x) \in \mathcal{C}_k[\![\theta(t)]\!]$ , by 4.1.2.

$\langle 1 \rangle 15$ . CASE: `TY_UNIT_INTRO`.

PROVE:  $(\sigma, \gamma(\delta(\text{() }))) \in \mathcal{C}_k[\![\theta(\mathbf{unit})]\!]$ .

$\langle 1 \rangle 16$ . CASE: `TY_BOOL_TRUE`, `TY_BOOL_FALSE`, `TY_INT_INTRO`, `TY_ELT_INTRO`.

Similar to `TY_UNIT_INTRO`.

## 5 Grammar Definition

$m$	$::=$		matrix expressions
		$M$	matrix variables
		$m + m'$	matrix addition
		$m \ m'$	matrix multiplication
		$(m)$	S
$f$	$::=$		fractional capability
		$fc$	variable
		$1$	whole capability
		$\frac{1}{2}f$	
$t$	$::=$		linear type
		<b>unit</b>	unit
		<b>bool</b>	boolean (true/false)
		<b>int</b>	63-bit integers
		<b>elt</b>	array element
		$f \text{ arr}$	arrays
		$f \text{ mat}$	matrices
		$!t$	multiple-use type
		$\forall fc.t$	bind $fc$ in $t$ frac. cap. generalisation
		$t \otimes t'$	pair
		$t \multimap t'$	linear function
		$(t)$	S    parentheses
$p$	$::=$		primitive
		<b>not</b>	boolean negation
		$(+)$	integer addition
		$(-)$	integer subtraction
		$(*)$	integer multiplication
		$(/)$	integer division
		$(=)$	integer equality
		$(<)$	integer less-than
		$(+.)$	element addition
		$(-.)$	element subtraction
		$(*.)$	element multiplication
		$(/.)$	element division
		$(=.)$	element equality
		$(<.)$	element less-than
		<b>set</b>	array index assignment
		<b>get</b>	array indexing
		<b>share</b>	share array
		<b>unshare</b>	unshare array
		<b>free</b>	free array
		<b>array</b>	Owl: make array
		<b>copy</b>	Owl: copy array
		<b>sin</b>	Owl: map sine over array
		<b>hypot</b>	Owl: $x_i := \sqrt{x_i^2 + y_i^2}$
		<b>asum</b>	BLAS: $\sum_i  x_i $

	<b>axpy</b>		BLAS: $x := \alpha x + y$
	<b>dot</b>		BLAS: $x \cdot y$
	<b>rotmg</b>		BLAS: see its docs
	<b>scal</b>		BLAS: $x := \alpha x$
	<b>amax</b>		BLAS: $\text{argmax } i : x_i$
	<b>setM</b>		matrix index assignment
	<b>getM</b>		matrix indexing
	<b>shareM</b>		share matrix
	<b>unshareM</b>		unshare matrix
	<b>freeM</b>		free matrix
	<b>matrix</b>		Owl: make matrix
	<b>copyM</b>		Owl: copy matrix
	<b>copyM_to</b>		Owl: copy matrix onto another
	<b>sizeM</b>		dimension of matrix
	<b>trnsp</b>		transpose matrix
	<b>gemm</b>		BLAS: $C := \alpha A^{T?} B^{T?} + \beta C$
	<b>symm</b>		BLAS: $C := \alpha AB + \beta C$
	<b>posv</b>		BLAS: Cholesky decomp. and solve
	<b>potrs</b>		BLAS: solve with given Cholesky
	<b>syrk</b>		BLAS: $C := \alpha A^{T?} A^{T?} + \beta C$
$v$	<b>::=</b>		values
	$p$		primitives
	$x$		variable
	$()$		unit introduction
	<b>true</b>		true
	<b>false</b>		false
	$k$		integer
	$l \cdot f$		heap location
	$el$		array element
	<b>Many</b> $v$		!-introduction
	<b>fun</b> $fc \rightarrow v$		frac. cap. abstraction
	$v[f]$		frac. cap. specialisation
	$(v, v')$		pair introduction
	<b>fun</b> $x : t \rightarrow e$	bind $x$ in $e$	abstraction
	<b>fix</b> $(g, x : t, e : t')$	bind $g \cup x$ in $e$	fixpoint
	$(v)$	S	parentheses
$e$	<b>::=</b>		expression
	$p$		primitives
	$x$		variable
	<b>let</b> $x = e$ <b>in</b> $e'$	bind $x$ in $e'$	let binding
	$()$		unit introduction
	<b>let</b> $() = e$ <b>in</b> $e'$		unit elimination
	<b>true</b>		true
	<b>false</b>		false
	<b>if</b> $e$ <b>then</b> $e_1$ <b>else</b> $e_2$		if
	$k$		integer

		$l \cdot f$		heap location
		$el$		array element
		<b>Many</b> $e$		!-introduction
		<b>let</b> <b>Many</b> $x = e$ <b>in</b> $e'$		!-elimination
		<b>fun</b> $fc \rightarrow e$		frac. cap. abstraction
		$e[f]$		frac. cap. specialisation
		$(e, e')$		pair introduction
		<b>let</b> $(a, b) = e$ <b>in</b> $e'$	<b>bind</b> $a \cup b$ <b>in</b> $e'$	pair elimination
		<b>fun</b> $x : t \rightarrow e$	<b>bind</b> $x$ <b>in</b> $e$	abstraction
		$e e'$		application
		<b>fix</b> $(g, x : t, e : t')$	<b>bind</b> $g \cup x$ <b>in</b> $e$	fixpoint
		$(e)$	<b>S</b>	parentheses
$C$	$::=$			evaluation contexts
		<b>let</b> $x = [-]$ <b>in</b> $e$	<b>bind</b> $x$ <b>in</b> $e$	let binding
		<b>let</b> $() = [-]$ <b>in</b> $e$		unit elimination
		<b>if</b> $[-]$ <b>then</b> $e_1$ <b>else</b> $e_2$		if
		<b>Many</b> $[-]$		!-introduction
		<b>let</b> <b>Many</b> $x = [-]$ <b>in</b> $e$		!-elimination
		<b>fun</b> $fc \rightarrow [-]$		frac. cap. abstraction
		$[-][f]$		frac. cap. specialisation
		$([-], e)$		pair introduction
		$(v, [-])$		pair introduction
		<b>let</b> $(a, b) = [-]$ <b>in</b> $e$	<b>bind</b> $a \cup b$ <b>in</b> $e$	pair elimination
		$[-]e$		application
		$v[-]$		application
$\Theta$	$::=$			fractional capability environment
		$\cdot$		
		$\Theta, fc$		
$\Gamma$	$::=$			linear types environment
		$\cdot$		
		$\Gamma, x : t$		
		$\Gamma, \Gamma'$		
$\Delta$	$::=$			intuitionistic types environment
		$\cdot$		
		$\Delta, x : t$		
$\sigma$	$::=$			heap (multiset of triples)
		$\{\}$		empty heap
		$\sigma + \{l \mapsto_f m_{k_1, k_2}\}$		location $l$ points to matrix $m$
$StepsTo$	$::=$			result of small step
		$\langle \sigma, e \rangle$		heap and expression
		<b>err</b>		error